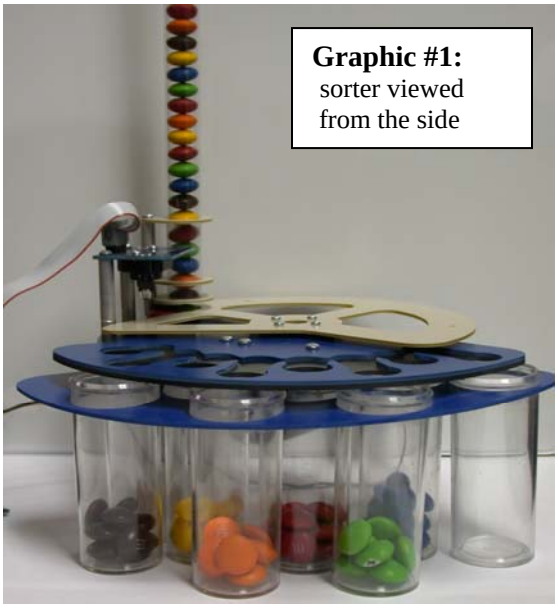
Simulating Manufacturing Processes in Education and Training: Enhanced Color Sorting Example

Richard Johnson
rjohnson@bsu.edu
Department of Technology
Ball State University

Introduction

This design brief describes enhancements to the project "Simulating Manufacturing Processes in Education and Training: Color Sorting Example" [1] previously published in the Spring 2006 edition of The Technology Interface. A review of that article is suggested as background study. The primary purpose of the original project was to simulate an industrial color sorting process on a scale suitable for use in the classroom, in this case sorting small M&M™ candies. The main components of the sorting system included a Parallax BS2™ microcontroller, a TCS230™ color sensor from Texas Advanced Optoelectronic Solutions, and a simple servo driven mechanical system (see graphic #1). Two key microcontroller programs were developed to operate the system; the first program dynamically calibrated the mechanical system and color sensor system, and the second program identified and sorted the candies into color groups. The original project design also required that a personal computer be attached to the sorting system for microcontroller program storage and for user input/output operations.



Graphic #1: sorter viewed from the side

The new project challenge was to enhance the current color sorting system, with a minimum of software changes, to be self contained and very portable. Making the sorting system portable required that the personal computer be replaced with a microcontroller driven user interface sub-system. The user interface functions of the personal computer were replaced with a small liquid crystal display (LCD) unit and simple push buttons mounted in a plastic project box (note graphic #2). Removing the personal computer from the system however, forced a microcontroller upgrade; the Parallax BS2™ microcontroller does not have enough memory to hold all of the programs needed to operate the color sorting system. The BS2™ microcontroller was upgraded to a Parallax BS2e™ microcontroller. The BS2e™ has an attribute that allows up to eight programs to be loaded concurrently into program "slots". A total of five programs are used with the enhanced color sorting system; a new main

menu or control program, the two previously mentioned calibration and sorting programs, plus two utility programs. Only minimal software flow control changes were required to implement the microcontroller upgrade.

User Interface

A small 4 line by 20 character serial liquid crystal display (LCD) was selected to replace the computer monitor for user output (note graphic #2). The selected LCD provided a good balance between cost and functionality. The small size (approximately 2.25" x 3.75") of the LCD unit was an advantage in making the sorting device portable, but a disadvantage due to the changes required to reformat the user output messages. While the number of software changes needed to reformat the user output messages were large, they were relatively easy to make. Generally the output messages were simply reworded, words abbreviated when possible, large displays were broken into smaller 4 line units, and input references changed from the PC keyboard to

Graphic #2: Project box (purchased from local Radio Shack™)

pushbuttons on the project box.

Originally the PBASIC command "DEBUG" was used to output user interface messages on the personal computer monitor. The LCD interface required the use of the PBASIC command "SEROUT" to transmit asynchronous serial data to the LCD display. A limited number of the

old DEBUG commands were retained in the new programs in the event the LCD display failed and the sorting system needed to be connected to a personal computer for diagnostic purposes

The Figures #1 and #2 show functionally equivalent program commands used to display the main system menu. Both examples first clear the display area and then show the characters in quotes beginning in the upper left corner of the display area. Figure #1 shows the DEBUG commands (PC version), while Figure #2 shows the SEROUT commands (LCD version).

Figure #1 - DEBUG example

```
DEBUG HOME, CLS
DEBUG "Operation      hit 1", CR
DEBUG "Calibration    hit 2", CR
DEBUG "Read EEPROM    hit 3", CR
DEBUG "Init EEPROM    hit 4", CR
```

Figure #2 - SEROUT example

```
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Operation      hit 1"]
SEROUT TX, LcdBaud, ["Calibration    hit 2"]
SEROUT TX, LcdBaud, ["Read EEPROM    hit 3"]
SEROUT TX, LcdBaud, ["Init EEPROM    hit 4"]
```

Details about each command can be found in the Basic Stamp Syntax and Reference Manual published by Parallax, Inc. (http://www.parallax.com).

Pushbutton switches were mounted in the project box to functionally replace the user input from the personal computer keyboard. Most of the user input via the pushbuttons is very straight forward, such as pushing the appropriate button to select a menu option or to continue an operation after an interruption. Five pushbuttons were installed in the project box, however, only four were used with this version of the color sorting system. The function of the toggle switch is to indicate that the sorting process should be stopped at the end of the next sorting cycle. The BS2e™ microcontroller does not have hardware interrupts which complicated the condition where the user wanted to manually stop the sorting process. Without hardware interrupts the sort program would need to test for a stop condition (pressed pushbutton) several times during the sorting cycle or the user would be required to hold a pushbutton down until the current sort cycle ended; neither of these options was acceptable to the system designer. Using the toggle switch simplified the situation where the user wanted to stop the sorting process prior to exhausting all of the candies in the supply tube. After the sorting process stops the user is prompted to reset the toggle switch before returning to the main system menu.

Microcontroller upgrade and program flow control

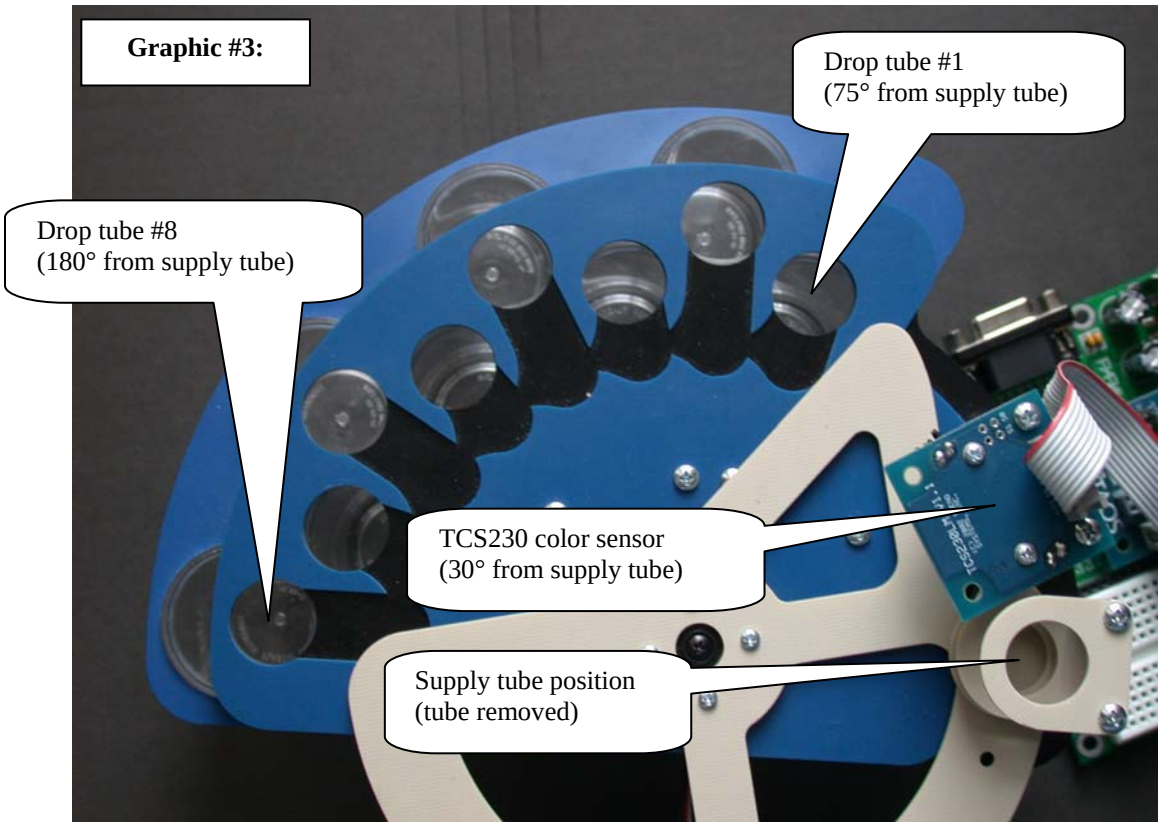
Upgrading from the original BS2™ microcontroller to the BS2e™ microcontroller in the same product family was central to making the color sorting system portable with a minimum of software changes. Originally the sorting system was attached to a personal computer that stored all of the programs required to operate the system, removing the personal computer from the system required that additional program storage space be available directly on the microcontroller. The BS2™ has only one block of 2048 bytes of program and data storage space (Electrically Erasable Programmable Read Only Memory or EEPROM). The BS2e™ has eight blocks of 2048 bytes of program and data storage space. Each 2048 byte block of storage space is called a “program slot”, identified as “program slot 0” through “program slot 7”. Program slot 0 is unique; when the microcontroller powers up or is reset, the program in slot 0 is executed. Since slot 0 is the default starting execution point, the system main menu program is loaded into this slot. The BS2e™ microcontroller also has 64 bytes of an additional type of

memory available called “scratch pad random access memory” or “SPRAM”. SPRAM is available to all of the program slots (global variables); in this sorting system the SPRAM is used to share data between the program slots.

Figure #3:
Program Slot Assignments
Slot 0: Main Menu
Slot 1: Sorting Program
Slot 2: Calibration Program
Slot 3: Read EEPROM Program
Slot 4: Init EEPROM Program

The following discussion will describe the software changes needed to adapt the original color sorting system programs to use the enhanced features of the BS2e™ microcontroller. The main system menu is the control point for the four other programs that are used with the enhanced color sorting system (note figure #3); sorting program, calibration program, and two utility programs. Most of the program changes needed for flow control are physically located at the program entry and exit points. Prior to transferring control to another program slot, the active program writes the current slot number to location 62 in the SPRAM. This allows the subsequent program to determine which program slot transferred control to it. For example, this technique allows the main menu program to distinguish between a power-on condition and the transfer of control from another program.

Graphic #3:



The first operations that must be completed are the mechanical calibration and color calibration of the sorting system by the calibration program. To review, the mechanical calibration routine determines the values needed to direct the servo motor to move the candies to the required positions (note graphic #3) on the sorting device.
The TCS230 sensor is used to locate the position of two alignment holes in the rotating plate, the servo positions for the supply tube and drop tubes can then calculated. The color calibration

routine determines the TCS230 sensor values that correlate to the six different colors of candies used in this project. Color pairs of candies are loaded into the supply tube in a specific sequence (brown, yellow, orange, red, green, and blue). Using the average sensor value from two candies of each color, the values are associated with the six different candy colors. The averaged sensor values are used in the sorting program to identify unknown candies. The values from both calibration routines are written to specific addresses in the slot #3 EEPROM space. Immediately before returning control to the main menu program, the calibration values are copied from the slot #3 EEPROM space to the SPRAM. Also SPRAM location #62 is updated with a value of "3" which will indicate to the main menu program that the calibration program has just executed and transferred control.

When control is returned to the main menu program it will update the local slot #0 EEPROM space with the modified values from the calibration program. When the color sorting system is powered-on or reset the main menu program will copy the calibration values into the SPRAM with the values previously stored in the slot #0 EEPROM. The duplication of calibration values in the two different EEPROM areas is to accommodate a future enhancement.

After the color sorting system is calibrated, it is ready to begin sorting the M&M™ candies. Before transferring control to the sorting program the main menu program checks the calibration flag (SPRAM address #8) to confirm that the sorting system has been successfully calibrated. From the program flow control perspective, entry into the sorting program required only one change; copy the SPRAM to the local slot #1 EEPROM. Within the main sorting program one addition was needed to check the status of the toggle switch on the project box. The toggle switch is used by the operator to stop the sorting process before the supply of candies is exhausted. As explained in the user interface section, using the toggle switch was a workaround compromise due to the lack of hardware interrupts on the BS2e microcontroller. Exiting from the sorting program required checking the status of the toggle switch and updating the SPRAM. If the toggle switch was flipped on by the operator to stop the sorting process, the operator is now prompted to reset the switch before being allowed to exit to the main menu. Immediately prior to returning control to the main menu program, SPRAM address #62 is updated with a value of "1", the slot number of the sorting program.

Two primary utility programs were used during (and after) the development of the color sorting system. The "Read_EEPROM" program was developed to display raw sensor data after a calibration run; it proved very useful in establishing the setup parameters for the TCS230 sensor. Significant changes were needed for this program due to moving the user output from a personal computer monitor to a 4 line by 20 character LCD. The output data was divided into groups of four lines that required the operator to press a button on the project box to scroll forward; no scroll backward function was written. Also, the program was updated to use only the SPRAM and not the EEPROM. Immediately prior to returning control to the main menu program SPRAM address #62 is updated with a value of "3", the slot number of the "Read_EEPROM" program. The "Init_EEPROM" program was written to manually set the servo motor positioning values; this program expedited the refinement of the servo positioning calculations. The numbers required to update the servo positioning values must be written directly into the program and the program recompiled. The flow control changes were needed at the program exit point; first eight bytes of the SPRAM were updated with the new servo positioning values. Only basic user interface changes were needed for this program. Prior to returning control to the main

menu program SPRAM address #62 is updated with a value of "4", the slot number of the "Init_EEPROM" program.

Summary

This design brief has described the first of several revisions to the M&M™ sorting device. Too often educational projects fall into the category of "use once and throw away". This author has in recent years developed a number of projects that challenges students to "design for change" and that requires students to enhance existing projects. The primary design goal for this project was portability, with an implied goal to enhance the color sorting system to be as self contained as possible.

Future enhancement projects will include both hardware and software components. The next software enhancement will be based on the technique of "conditional compilation"; this type of update will demonstrate how to develop program code for different microcontrollers in the same family while conserving memory. The next hardware enhancements will extend the user interface to the World Wide Web through the use of board level web servers and streaming video. The enhanced user interface will allow the M&M™ sorting device to be monitored and to some extent remotely controlled through the Web.

Appendix "A" lists all the microcontroller programs used with this project.

Acknowledgements

The author would like to thank Mr. Christopher A. Hileman for his technical contributions and dedication to the development of the original project and Mr. Luke Jamison for his technical leadership to this first enhanced version of the M&M™ sorting device.

References and Materials

[1] Johnson, R. & Cotton, S. (2006). Simulating Manufacturing Processes in Education and Training: Color Sorting Example, the Technology Interface, 3(1)

Parallax, Inc. (2008). "Basic Stamp Syntax and Reference Manual", version 2.2.
Available at http://www.parallax.com.

Radio Shack (2008). Project enclosure. Product #270-1809
Available at http://www.radioshack.com.

Parallax, Inc. (2008). 4x20 Serial LCD. Product #27979
Available at http://www.parallax.com.

Appendix A – Program Listing

This appendix contains five (5) microcontroller program listings used with the color sorting device.

- Main menu program listing
- Calibration program listing
- Sorting program listing
- ReadEEPROM utility program listing
- InitEEPROM utility program listing

Main menu program listing

```
' {$STAMP BS2e, Sorting_Program, Calibration_Program, ReadEEPROM,
Init_EEPROM}
' {$PBASIC 2.5}
'
'************************************************************
'  Program: Main Menu for BS2e controller
'  Purpose: To switch between slots to different programs based to users
choice
'  Authors: Rick Johnson & Luke Jamison
'  Date:    April 2007
'************************************************************

TX              PIN     9               ' serial output to LCD

    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32
LcdBaud         CON     T19K2

LcdBkSpc        CON     $08             ' move cursor left
LcdRt           CON     $09             ' move cursor right
LcdLF           CON     $0A             ' move cursor down 1 line
LcdCls          CON     $0C             ' clear LCD (use PAUSE 5 after)
LcdCR           CON     $0D             ' move pos 0 of next line
LcdBLon         CON     $11             ' backlight on
LcdBLoff        CON     $12             ' backlight off
LcdOff          CON     $15             ' LCD off

Cal_flag        CON       8

Menu_pgm        CON       0             ' Program slot numbers
Sort_pgm        CON       1
Cal_pgm         CON       2
Read_pgm        CON       3
Init_pgm        CON       4
SPRAM_Lastslot  CON      62

Toggleswitch    PIN       0
Button_1        PIN      10             'pushbutton pins
Button_2        PIN      13
```

```
Button_3        PIN      14
Button_4        PIN      15
'Button_5       PIN      11                not used in programs yet

Answer          VAR    Byte
temp            VAR    Byte
temp_2          VAR    Byte
Mem_Loc         VAR    Byte



Initchk:                                    'checks where the program came
from
GET SPRAM_Lastslot, temp
IF temp = 0 THEN                            ' power on condition
    GOSUB write_spram                       ' load SPRAM from local EEPROM
ELSEIF (temp = Cal_pgm) OR (temp = Init_pgm) THEN  ' returning from
calibration or
  GOSUB SPRAM_to_EEPROM                        ' Init_EEPROM program.
ENDIF                                          ' Update local EEPROM.


Menu:
PAUSE 1000
SEROUT TX, LcdBaud, [LcdCls]
SEROUT TX, LcdBaud, [LcdBLoff]           'turning backlight off
PAUSE 250
SEROUT TX, LcdBaud, ["Operation      hit 1"]
SEROUT TX, LcdBaud, ["Calibration    hit 2"]
SEROUT TX, LcdBaud, ["Read EEPROM    hit 3"]
SEROUT TX, LcdBaud, ["Init EEPROM    hit 4"]

DEBUG HOME, CLS
DEBUG "Operation      hit 1", CR
DEBUG "Calibration    hit 2", CR
DEBUG "Read EEPROM    hit 3", CR
DEBUG "Init EEPROM    hit 4", CR

DO                                  ' Wait until a menu button is pressed
   PAUSE 1
LOOP WHILE ((BUTTON_1 = 0) AND (Button_2 = 0) AND (Button_3 = 0) AND
(Button_4 = 0))

IF (Button_1 = 1) THEN
   GOTO  Operation
ELSEIF (Button_2 = 1) THEN
   GOTO Calibration
ELSEIF (Button_3 = 1) THEN
   GOTO READ_EEPROM
ELSEIF (Button_4 = 1) THEN
   GOTO Init_eeprom
ELSE
  GOTO Menu
ENDIF


Operation:
```

```
GET Cal_flag, temp                          ' Confirm system is calibrated
IF temp = 199   THEN
   PUT SPRAM_Lastslot, Menu_pgm
   SEROUT TX, LcdBaud, [LcdCls]
   PAUSE 5
   RUN Sort_pgm
ENDIF

SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Not Calibrated"]
DEBUG HOME, CLS, "Not Calibrated"
PAUSE 3000
GOTO menu


Calibration:
PUT SPRAM_Lastslot, Menu_pgm
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 5
RUN Cal_pgm


READ_EEPROM:
PUT SPRAM_Lastslot, Menu_pgm
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 5
RUN Read_pgm


INIT_EEPROM:
PUT SPRAM_Lastslot, Menu_pgm
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 5
RUN Init_pgm
'-----------------------------------------


SPRAM_to_EEPROM:                    'checking to make sure both EEPROM and
SPRAM are the same
FOR Mem_Loc = 0 TO 58
    GET Mem_Loc, temp
    READ Mem_Loc, temp_2
    IF temp <> temp_2 THEN
       WRITE Mem_Loc, temp
    ENDIF
  NEXT
RETURN


write_SPRAM:
FOR Mem_Loc = 0 TO 58               'puts local EEPROM into SPRAM
    READ Mem_Loc, Temp
    PUT Mem_Loc, Temp
NEXT
RETURN
```

Calibration program listing

```
'{$STAMP BS2e}
'{$PBASIC 2.5}
'*********************************************'
'     Program = Operation_Program              '
'     Purpose:  Use this program to calibrate  '
'      the Parallax M&M Sorter                 '
'     Authors:  Rick Johnson & Chris Hileman   '
'     Date: January 2004                       '
'     Modified by: Rick Johnson & Luke Jamison '
'                  April 2007                  '
'*********************************************'

'LCD declarations
TX              PIN     9               ' serial output to LCD

    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32

LcdBaud         CON     T19K2

LcdBkSpc        CON     $08              ' move cursor left
LcdRt           CON     $09              ' move cursor right
LcdLF           CON     $0A              ' move cursor down 1 line
LcdCls          CON     $0C              ' clear LCD (use PAUSE 5 after)
LcdCR           CON     $0D              ' move pos 0 of next line
LcdBLon         CON     $11              ' backlight on
LcdBLoff        CON     $12              ' backlight off
LcdOff          CON     $15              ' LCD off


Menu_pgm        CON       0              ' Program slot numbers
Sort_pgm        CON       1
Cal_pgm         CON       2
Read_pgm        CON       3
Init_pgm        CON       4

Button_1        PIN      10              'pushbutton pins
Button_2        PIN      13
Button_3        PIN      14
Button_4        PIN      15

'*************************
'Program Declarations
'*************************
Servo_pin   CON   12
EN          CON   1
A0          CON   2
S0          CON   3
S1          CON   4
S2          CON   5
S3          CON   6
```

```
nLED        CON   7
OUT         CON   8

pRED        CON   12
pGREEN      CON   8
pBLUE       CON   12

RED         VAR   Word
GREEN       VAR   Byte
BLUE        VAR   Byte
Last_RED    VAR   Word
Last_GREEN  VAR   Byte
Last_BLUE   VAR   Byte
'************************
'Calulation Declarations
'************************
Align_Hole_2  VAR Word                  'Calibration Hole2 VAR
Align_Hole_1  VAR Word                  'Calibration Hole1 VAR
Target_Pos    VAR Word                  'Location Calculation var
ColorCal      VAR Word                  'Color Calculation VAR
Cur_pos       VAR Word                  'Current Position VAR
Next_pos      VAR Word                  'Next Position VAR
ColorCal2     VAR Word                  'Color Calculation VAR

Pos_Delta     VAR Byte                  'Delta of Position Increments
Cur_Color     VAR Byte                  'Current Color
Home_Pos      VAR Byte                  'Home Position
Sensor_Pos    VAR Align_Hole_2          'Reuse of Variable for Sensor
Position
Hole_1        VAR Align_Hole_1          'Reuse of Variable for Drop Hole #1

Cal_Flag      CON 8

Answer        VAR Nib                   'Menu Choice (DEBUGIN)
Cal           VAR Bit                   'Calibration flag  on / off

BaseR         CON 10                    'EEPROM ADDRESS
FinishR       CON 5 * 8 + BaseR         'EEPROM ADDRESS
'SPRAM_High    CON 62
SPRAM_Lastslot  CON 62
'************************
'Scratchpad Copying Declarations
'************************
Mem_Loc       VAR Last_Green             'Reusing Last_Green
TEMP          VAR Last_Blue              'Reusing Last_Blue


FOR Mem_Loc = 0 TO 58                    ' update local EEPROM from SPRAM
    GET Mem_Loc, TEMP
    WRITE  Mem_Loc, TEMP
NEXT

Cal = 0
READ Cal_Flag, Cur_Color                          'current color = temp var
IF Cur_Color = 99 OR Cur_Color = 199 THEN
  Cal = 1
ENDIF
```

```
'*************************************************
Start:                                          'Start loop for Calibration
User interface menus
Answer = 0
DEBUG CLS, "Welcome to Calibration", CR, CR
'DEBUG "What would you like to do? ", CR
DEBUG "[1] Calibrate Control Arm", CR
SEROUT TX, LcdBaud, [LcdCls]

PAUSE 250
SEROUT TX, LcdBaud, ["Calibrate Arm hit 1"]
SEROUT TX, LcdBaud, [LcdCR]

IF Cal = 1 THEN
  DEBUG "[2] Calibrate Color Sensor", CR
  SEROUT TX, LcdBaud, ["Calibrate Color 2"]
  SEROUT TX, LcdBaud, [LcdCR]
ELSE
  DEBUG " ", CR
ENDIF

DEBUG "[3] Main Menu", CR
SEROUT TX, LcdBaud, ["Main Menu 3"]
SEROUT TX, LcdBaud, [LcdCR]
    'DEBUGIN DEC1 Answer
    'DEBUG CR, DEC1 Answer
    'BRANCH Answer, [Start, Arm_Calibration, Color_Calibration, Main_Menu,
Start]
    'GOTO Start
DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0) AND (Button_2 = 0) AND (Button_3 = 0))

IF (Button_1 = 1) THEN
    GOTO Arm_Calibration
ELSEIF (Button_2 = 1) THEN
    GOTO Color_Calibration
ELSEIF (Button_3 = 1) THEN
    GOTO Main_Menu
ENDIF


'*********************************
Arm_Calibration:                                'Start of Arm Calibration
routine
  Cur_pos = 800
  Answer  = 1
Arm_Cal_Loop:
DEBUG CLS, "Arm Alignment", CR
DEBUG CR, "Press '1' until arm is fully rotated then press '2'", CR
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Tap 1 until arm is"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["fully rotated, then"]
SEROUT TX, LcdBaud, [LcdCR]
```

```
SEROUT TX, LcdBaud, ["press 2."]

DO WHILE (Answer = 1)
  Next_pos = Cur_pos + 100
  FOR Cur_pos = Cur_pos TO Next_pos STEP 5
    PULSOUT Servo_pin, Cur_pos
'*************************************'
    PAUSE 10                                          'Press 1 until the arm
is rotated, this  '
  NEXT                                                'to calibrate the arm
starting with the  '
                                                      'location of calibration
hole #2 and     '
  DO                                                  'then moving to hole #1
'
    PAUSE 1
'*************************************'
  LOOP WHILE ((Button_1 = 0) AND (Button_2 = 0))

  IF (Button_1 = 1) THEN
        Answer = 1
  ELSEIF (Button_2 = 1) THEN
        Answer = 2
  ENDIF
LOOP

Next_pos = Cur_pos - 200
GOSUB Check_Color
Last_RED   = RED
Last_GREEN = GREEN
Last_BLUE  = BLUE
FOR Cur_pos = Cur_pos TO Next_pos STEP 2
  PULSOUT Servo_pin, Cur_pos
  PAUSE 10
  GOSUB Check_Color
  IF (RED <= Last_RED + 1) AND (GREEN <= Last_GREEN + 1) AND (BLUE <=
Last_BLUE + 1) THEN
    Last_RED   = RED
    Last_GREEN = GREEN
    Last_BLUE  = BLUE
  ELSE
'***********************************************'
    Align_Hole_2 = Cur_pos                        'In this loop, it records
the previous RGB values '
    EXIT                                          'and then compares those
versus the present values'
  ENDIF                                           'If there is any drop in
Number, (A Hole Located) '
NEXT                                              'That location is then
recorded for future use in '

FOR Cur_pos = cur_pos TO 500 STEP 5               'other calculations that
are to be done           '
  PULSOUT Servo_pin, Cur_pos
'***********************************************'
  PAUSE 10
NEXT
```

```
Next_pos = Cur_pos - 200
GOSUB Check_Color
Last_RED   = RED
Last_GREEN = GREEN
Last_BLUE  = BLUE
FOR Cur_pos = Cur_pos TO 200 STEP 2
  PULSOUT Servo_pin, Cur_pos
  PAUSE 30
  GOSUB Check_Color
  IF (RED <= Last_RED + 2) AND (GREEN <= Last_GREEN + 2) AND (BLUE <=
Last_BLUE + 2) THEN
    Last_RED   = RED
    Last_GREEN = GREEN
    Last_BLUE  = BLUE
  ELSE
    Align_Hole_1 = Cur_pos
    EXIT
  ENDIF
NEXT


Compute_Target_Locations:                                    'Loop Calculates
Primary Locations
Pos_Delta = Align_hole_2 - Align_hole_1 * 10 / 8 + 5 / 10
WRITE 0, Word Pos_delta                                      ' Write to
EEPROM Position Delta
Target_Pos = Align_Hole_1 - (Pos_delta * 2)
WRITE 4, Word Target_pos                                     ' Write to
EEPROM Sensor Position
Target_Pos = Align_Hole_1 - (Pos_delta * 4) + 22
WRITE 6, Word Target_pos                                     ' Write to
EEPROM Tube/Home Position
Target_Pos = Align_Hole_1 + Pos_delta
WRITE 2, Word Target_pos                                     ' Write to
EEPROM Drop Hole #1 Position
IF cal = 0 THEN
  WRITE Cal_Flag, 99
ENDIF
Cal = 1

GOTO Start


'**************************************
Check_Color:                  ' Read the color of subject with sensor
  LOW A0                      ' Init Sensor
  HIGH  S0
  HIGH  S1
  LOW nLED                    ' LED's On
  PAUSE 200
  HIGH EN
  LOW S2                      ' Test for color
  LOW S3
  COUNT out, pRED, RED
  HIGH S3
  COUNT out, pBLUE, BLUE
```

```
  HIGH S2
  COUNT OUT, pGREEN, GREEN
  LOW EN                        ' Sensor off
  LOW nLED                      ' LED's Off
RETURN


'****************************
'************************************'
Color_Calibration:                                      'Test to see if Arm
has been Calibrated'
IF Cal = 0 THEN                                         'and then Calibrate
color Sensor follow'
  DEBUG CR, "Run Arm Calibration", CR                   'On screen
instructions                '
  PAUSE 5000
'************************************'
  GOTO Start
ENDIF

READ 0, Pos_delta
READ 2, Word Hole_1
READ 4, Word Sensor_Pos
READ 6, Home_Pos

'DEBUG CR, "Load test M&M's into tube ",CR
DEBUG CR, "Load M&M's",CR
DEBUG "Brown, Yellow, Orange, Red, Green, Blue ",CR
DEBUG "Press 1 to continue ",CR
'DEBUGIN DEC1 Answer
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Load Test M&M's"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Br Y Or R Gr Bl"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Press 1 to continue"]

DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))

FOR Cur_Color = BaseR TO FinishR STEP 8
  FOR Answer = 1 TO 2 STEP 1
    GOSUB Sensor_Position
    GOSUB Check_Color
    IF Answer = 1 THEN
      Last_Red   = Red
      Last_Blue  = Blue
      Last_Green = Green
'****************************************'
    ELSE                                                 'Uses colors that
the sensor detects and  '
      IF ((Last_Red + Red) / 2) > 100 THEN               'Calculates ranges
of the colors that we  '
        ColorCal   = (Last_Red + Red) * 9 / 20           'will use to compare
the sensor's readings'
```

```
        ColorCal2  = (Last_Red + Red) * 11 / 20          'later.  It then
writes those ranges to   '
      ELSE                                              'the proper EEPROM
address               '
        ColorCal   = (Last_Red + Red) * 8 / 20
'*************************************'
        ColorCal2  = (Last_Red + Red) * 12 / 20
      ENDIF
      WRITE Cur_Color, Word ColorCal
      WRITE (Cur_Color + 2), Word ColorCal2

      IF ((Last_Green + Green) / 2) > 100 THEN
        ColorCal   = (Last_Green + Green) * 9 / 20
        ColorCal2  = (Last_Green + Green) * 11 / 20
      ELSE
        ColorCal   = (Last_Green + Green) * 7 / 20
        ColorCal2  = (Last_Green + Green) * 13 / 20
      ENDIF
      WRITE (Cur_Color + 4), ColorCal
      WRITE (Cur_Color + 5), ColorCal2

      IF ((Last_Blue + Blue) / 2) > 100 THEN
        ColorCal   = (Last_Blue + Blue) * 9 / 20
        ColorCal2  = (Last_Blue + Blue) * 11 / 20
      ELSE
        ColorCal   = (Last_Blue + Blue) * 7 / 20
        ColorCal2  = (Last_Blue + Blue) * 13 / 20
      ENDIF
      WRITE (Cur_Color + 6), ColorCal
      WRITE (Cur_Color + 7), ColorCal2

      PAUSE 20
    ENDIF
    GOSUB Brown_Pos
  NEXT
NEXT

WRITE Cal_flag, 199

Cal = 1
GOSUB Sensor_Position
DEBUG CR, "Press '1' to continue "
'DEBUGIN DEC1 answer
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Hit 1 to continue"]
DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))
GOTO Start


Sensor_Position:                                    'Loop to position Servo From
HOME -> Sensor
  FOR Cur_pos = Home_Pos TO Sensor_Pos STEP 2
    PULSOUT Servo_pin, Cur_pos
    PAUSE 20
```

```
  NEXT
RETURN


Brown_Pos:                                          'Arbitray Dump for all
M&M's (Hole #1)
  FOR Cur_pos = Sensor_pos TO Hole_1 STEP 4
    PULSOUT Servo_Pin, Cur_Pos
    PAUSE 20
  NEXT
     FOR Cur_pos = Hole_1 TO Sensor_Pos STEP 25   'Drop Hole to Sensor Fast
return
      PULSOUT Servo_Pin, Cur_pos
      PAUSE 20
     NEXT
  GOSUB Check_Color
  FOR Cur_pos = Sensor_pos TO Home_Pos STEP 4     'To allow servo to return
from Sensor -> Home slowly
    PULSOUT Servo_Pin, Cur_pos
    PAUSE 20
  NEXT
 RETURN


'***************************** GO TO MAIN MENU ******************************
Main_Menu:
FOR Mem_Loc = 0 TO 58                               ' read EEPROM, writing
Scratchpad before Main Menu
    READ Mem_Loc, TEMP
    PUT  Mem_Loc, TEMP
NEXT

READ Cal_flag, temp
PUT SPRAM_Lastslot, Cal_pgm
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 5
RUN Menu_pgm
```

Sorting program listing

```
'{$STAMP BS2e}
'{$PBASIC 2.5}

'********************************************'
'     Program = Sorting_Program                '
'     Purpose:  Use this program to sort       '
'      M&M's using the Parallax M&M Sorter     '
'     Authors:  Rick Johnson & Chris Hileman   '
'     Date: January 2004                       '
'     Modified by: Rick Johnson & Luke Jamison '
'                  April 2007                  '
'                                              '
'********************************************'
'LCD Declarations
TX              PIN     9              ' serial output to LCD
```

```
    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32
LcdBaud         CON     T19K2

LcdBkSpc        CON     $08             ' move cursor left
LcdRt           CON     $09             ' move cursor right
LcdLF           CON     $0A             ' move cursor down 1 line
LcdCls          CON     $0C             ' clear LCD (use PAUSE 5 after)
LcdCR           CON     $0D             ' move pos 0 of next line
LcdBLon         CON     $11             ' backlight on
LcdBLoff        CON     $12             ' backlight off
LcdOff          CON     $15             ' LCD off

Menu_pgm        CON       0             ' Program slot numbers
Sort_pgm        CON       1
Cal_pgm         CON       2
Read_pgm        CON       3
Init_pgm        CON       4

Toggleswitch    PIN      0
Button_1        PIN      10             'pushbutton pins
Button_2        PIN      13
Button_3        PIN      14
Button_4        PIN      15

'*************************
'Program Declarations
'*************************
Sensor_Pos      VAR   Word          'Sensor_Pos = Position of the RGB Sensor
Hole_1          VAR   Word          'Pos_1 = Position of the #1 Tube
Pos_Delta       VAR   Byte          'Delta of the Position of the tube
Home_Pos        VAR   Byte          'Home_pos = Home position of Servo

RedHigh         VAR   Word          'High Red VAR
RedLow          VAR   Word          'Low Red VAR
Cur_x           VAR   Word          'Cur_X = current position of Servo
Word_Var        VAR   Word          'Variable used for different
calculations in program
GreenHigh       VAR   Byte          'High Green VAR
GreenLow        VAR   Byte          'Low Green VAR
BlueHigh        VAR   Byte          'High Blue VAR
BlueLow         VAR   Byte          'Low Blue VAR

Mem_Loc         VAR   BlueHigh      'reusing variable
TEMP            VAR   BlueLow


mmColor         VAR   Nib           'Variable set to signal M&M Color
Counter         VAR   Nib           'Counter Variable
A               VAR   Bit           'Variable used for multiple things

Servo_Pos       CON   12
mmColorUnknown  CON   6             'To tell Branch statement unknown Color
Move_MM_Step    CON   8             'Step Value for all Branch cases
```

```
'****************************
'EEPROM ADDRESSES
'****************************
BaseR           CON   10
FinishR         CON   5 * 8 + BaseR

'SPRAM_High    CON 62
SPRAM_Lastslot  CON 62
Cal_flag        CON 8

'****************************
'Color Sensor Declarations
'****************************
EN              CON   1
A0              CON   2
S0              CON   3
S1              CON   4
S2              CON   5
S3              CON   6
nLED            CON   7
OUT             CON   8

pRED            CON   12
pGREEN          CON   8
pBLUE           CON   12

RED             VAR   Word
GREEN           VAR   Word
BLUE            VAR   Word
'****************************
'End of Declarations
'****************************
FOR Mem_Loc = 0 TO 58                   'Writing Scratchpad to EEPROM
   GET Mem_Loc, TEMP                    ' reusing variables
   WRITE Mem_Loc, TEMP
NEXT

READ Cal_flag, TEMP                     'Checking for Calibration of Sorter
IF TEMP <> 199 THEN
  GOTO Check_Cal
ENDIF

READ 0, Pos_Delta                       'EEPROM Read for Variables
READ 2, Word Hole_1                     'EEPROM Read for Variables
READ 4, Word Sensor_Pos                 'EEPROM Read for Variables
READ 6, Home_Pos                        'EEPROM Read for Variables


MainStart:                              'Debug Menu for Start of Program
DEBUG REP "*" \40, CR
DEBUG CR, "Welcome to the M&M Sorter, "
DEBUG CR, "Load with M&M's and press 1",CR, CR
DEBUG REP "*" \40, CR

SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Load tube and hit 1"]
```

```
'  DEBUGIN DEC1 A
DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))


InnerMain:
'
'   check switch to end prog, and exit to main menu
IF toggleswitch = 1 THEN
   GOTO Main_Menu
ENDIF

'****************************************'
GOSUB Sensor_Position                    'This is the Main control loop for
this   '
DEBUG CR, "Checking Color",CR            'program.  It uses multiple sub-
routines  '
                                         'to accomplish this
'
SEROUT TX, LcdBaud, [LcdCls]
'****************************************'
PAUSE 250
SEROUT TX, LcdBaud, ["Checking color"]
SEROUT TX, LcdBaud, [LcdCR]
GOSUB Check_color
DEBUG " Now Placing M&M"
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Placing M&M"]
SEROUT TX, LcdBaud, [LcdCR]
GOSUB Color_Choice
GOSUB Drop_mm
IF Counter > 1 THEN                      'Will check supply tube twice to see
if its empty
    Counter = 0
    DEBUG CR, "Intervention Required", CR
    DEBUG "Press 1 and Enter to restart", CR
    DEBUG "Press 2 and Enter for Main Menu", CR, "--> "
    SEROUT TX, LcdBaud, [LcdCls]
    SEROUT TX, LcdBaud, [LcdBLon]
    PAUSE 250
    SEROUT TX, LcdBaud, ["Error, load tube to"]
    SEROUT TX, LcdBaud, [LcdCR]
    SEROUT TX, LcdBaud, ["restart, and hit 1"]
    SEROUT TX, LcdBaud, [LcdCR]
    SEROUT TX, LcdBaud, ["or Main Menu hit 2"]
    SEROUT TX, LcdBaud, [LcdCR]
    DO
      PAUSE 1
    LOOP WHILE ((BUTTON_1 = 0) AND (Button_2 = 0))
    IF (Button_1 = 1) THEN
      GOTO MainStart
    ELSEIF (Button_2 = 1) THEN
      GOTO Main_menu
    ENDIF                        'Done sorting
```

```
ENDIF
GOTO Innermain


Check_color:
'****************************************'
  HIGH EN                                 'Loop will use color sensor to check
the  '
  LOW A0                                  'color and then store the values it
sees  '
  HIGH  S0
'****************************************'
  HIGH  S1
  LOW nLED
  PAUSE 200
  LOW S2
  LOW S3
  COUNT OUT, pRED, RED
  HIGH S3
  COUNT OUT, pBLUE, BLUE
  HIGH S2
  COUNT OUT, pGREEN, GREEN
  LOW EN
  LOW nLED
RETURN


Color_Choice:
'************************************'
IF (RED < 10) AND (GREEN < 10) AND (BLUE < 10) THEN    'Loop gets information
and then decides'
  mmColor = 15                                         'Whether it is a known
color or unknown'
  RETURN                                               'Then uses lookdown
command to set a   '
ENDIF                                                  'Value to a Variable
for future use    '
mmColor = mmColorUnknown
'************************************'
FOR Word_Var = BaseR TO FinishR STEP 8
  READ Word_Var, Word RedLow
  READ (Word_Var + 2), Word RedHigh
  READ (Word_Var + 4), GreenLow
  READ (Word_Var + 5), GreenHigh
  READ (Word_Var + 6), BlueLow
  READ (Word_Var + 7), BlueHigh
  IF (RED >= RedLow) AND (RED <= RedHigh) AND (GREEN >= GreenLow) AND (GREEN
<= GreenHigh) AND (BLUE >= BlueLow) AND (BLUE <= BlueHigh) THEN
    LOOKDOWN Word_Var, [BaseR, BaseR + 8, BaseR + 16, BaseR + 24, BaseR + 32,
BaseR + 40, BaseR + 48], mmColor
    DEBUG CR ,"Found Color"
    SEROUT TX, LcdBaud, [LcdCls]
    PAUSE 250
    SEROUT TX, LcdBaud, ["Found Color"]
    SEROUT TX, LcdBaud, [LcdCR]
    EXIT
  ENDIF
```

```
NEXT
RETURN


'*****************************************'
Drop_mm:                                    'Loop using Branch statement
for tube location'
IF mmColor = 15 THEN                        'of the M&M and then sends a
signal to the    '
    Counter = Counter + 1                   'proper subroutine to sort the
M&M            '
    GOTO Drop_mm_Exit
'*****************************************'
ENDIF
DEBUG "Now Sorting M&M",CR
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Now Sorting M&M"]
SEROUT TX, LcdBaud, [LcdCR]
Cur_x = Sensor_Pos
BRANCH mmColor, [Brown_Pos, Yellow_Pos, Orange_Pos, Red_Pos, Green_Pos,
Blue_Pos, Unknown_Pos]
DEBUG "ERROR!", CR
DEBUG "Attention Required!", CR
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["CRITICAL ERROR!"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Attention Required!"]
GOTO Drop_mm_Exit


Unknown_Pos:                                'Tube location for Unknown
Color
FOR Cur_x = Cur_x TO Hole_1 STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
NEXT
GOTO Moveit


Brown_Pos:                                  'Tube location for Brown
M&M
Word_Var = Hole_1 + Pos_Delta
FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
NEXT
GOTO Moveit


Yellow_Pos:                                 'Tube location for Yellow
M&M
Word_Var = Hole_1 + (2 * Pos_Delta)
FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
```

```
NEXT
GOTO Moveit


Orange_Pos:                                          'Tube location for Orange
M&M
Word_Var = Hole_1 + (3 * Pos_Delta)
FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
NEXT
GOTO Moveit


Red_Pos:                                             'Tube location for Red M&M
Word_Var = Hole_1 + (4 * Pos_Delta)
FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
NEXT
GOTO Moveit


Green_Pos:                                           'Tube location for Green
M&M
Word_Var = Hole_1 + (5 * Pos_Delta)
FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
NEXT
GOTO Moveit


Blue_Pos:                                            'Tube location for Blue M&M
Word_Var = Hole_1 + (6 * Pos_Delta)
FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
NEXT
GOTO Moveit


Moveit:
'*********************************'
FOR Cur_x = Cur_x TO Sensor_Pos STEP 30            'Subroutine that moves
servo back '
  PULSOUT Servo_Pos, Cur_x                                 'to the sensor
position fast and  '
  PAUSE 20                                         'and then Sensor -> Home
slowly    '
NEXT
'*********************************'
Cur_x = Cur_x + 30
GOSUB Check_Color
PAUSE 500
GOSUB Home_Position
```

```
Drop_mm_Exit:
RETURN


Sensor_Position:                              'Loop to position Servo From HOME
-> Sensor
FOR Cur_x = Home_Pos TO Sensor_Pos STEP 2
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
NEXT
RETURN


Home_Position:                                'Loop Gets Servo from Current
location to Home
FOR Cur_x = Cur_x TO Home_Pos STEP 4
   PULSOUT Servo_Pos, Cur_x
   PAUSE 20
NEXT
RETURN


Check_Cal:                                    'Loop to instruct user to
calibrate M&M sorter before using
DEBUG CR, REP "*" \40 , CR
DEBUG "Go to Calibration Prog", CR
DEBUG "to Calibrate this Sorter", CR
DEBUG REP "*" \40 , CR
DEBUG "Main Menu press Enter", CR
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Please Calibrate"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Main Menu hit 1"]

DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))
GOTO Main_Menu


'************************ Main Menu ***************************
Main_Menu:
IF Toggleswitch = 1 THEN
   SEROUT TX, LcdBaud, [LcdCls]
   PAUSE 250
   SEROUT TX, LcdBaud, ["Please flip toggle"]
   SEROUT TX, LcdBaud, [LcdCR]
   SEROUT TX, LcdBaud, ["switch to continue"]
   SEROUT TX, LcdBaud, [LcdCR]
   SEROUT TX, LcdBaud, ["then hit 1"]
   DO
     PAUSE 1
   LOOP WHILE ((BUTTON_1 = 0))
ENDIF

IF toggleswitch = 1 THEN
```

```
   GOTO Main_menu
ENDIF

PUT SPRAM_Lastslot, Sort_pgm
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 5

RUN Menu_pgm               ' return to main menu
END
```

Read EEPROM Program listing

```
'{$STAMP BS2e}
'{$PBASIC 2.5}
'*****************************************
'  Program: ReadEEPROM  (utility program)
'  Purpose: utility program to display the contents
'           of memory on the LCD & PC monitor
'  Authors: Rick Johnson & Chris Hileman
'  Date:    January 2004
'  Modified by: Rick Johnson & Luke Jamison
'               April 2007
'*****************************************

TX              PIN     9              ' serial output to LCD

    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32

LcdBaud         CON     T19K2

LcdBkSpc        CON     $08             ' move cursor left
LcdRt           CON     $09             ' move cursor right
LcdLF           CON     $0A             ' move cursor down 1 line
LcdCls          CON     $0C             ' clear LCD (use PAUSE 5 after)
LcdCR           CON     $0D             ' move pos 0 of next line
LcdBLon         CON     $11             ' backlight on
LcdBLoff        CON     $12             ' backlight off
LcdOff          CON     $15             ' LCD off

Menu_pgm        CON       0             ' Program slot numbers
Sort_pgm        CON       1
Cal_pgm         CON       2
Read_pgm        CON       3
Init_pgm        CON       4
SPRAM_Lastslot  CON      62
Cal_flag        CON       8

Button_1        PIN      10             'pushbutton pins
Button_2        PIN      13
Button_3        PIN      14
Button_4        PIN      15

Word_Var        VAR   Word
```

```
BaseR           CON   10
FinishR         CON   5 * 8 + BaseR
RedHigh         VAR   Word
RedLow          VAR   Word
GreenHigh       VAR   Byte
GreenLow        VAR   Byte
BlueHigh        VAR   Byte
BlueLow         VAR   Byte
Pos_Vars        VAR   Word
answer          VAR   Byte


DEBUG CLS
GET 0, Pos_Vars
DEBUG "Arm - Pos_Delta = ", DEC Pos_Vars, CR
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Pos_Delta = ", DEC Pos_Vars]
SEROUT TX, LcdBaud, [LcdCR]

GET 2, Word Pos_Vars
DEBUG "Arm - Hole #1   = ", DEC Pos_Vars, CR
SEROUT TX, LcdBaud, ["Hole #1 = ", DEC  Pos_Vars]
SEROUT TX, LcdBaud, [LcdCR]

GET 4, Word Pos_Vars
DEBUG "Arm - Sensor    = ", DEC Pos_Vars, CR
SEROUT TX, LcdBaud, ["Sensor = ", DEC  Pos_Vars]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["hit 1 to scroll #'s"]
DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))


GET 6, Pos_Vars
DEBUG "Arm - Home      = ", DEC Pos_Vars, CR
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Home = ", DEC  Pos_Vars]
SEROUT TX, LcdBaud, [LcdCR]

GET  Cal_flag, Pos_Vars
DEBUG "Cal. Flag = ", DEC Pos_Vars, CR, CR
SEROUT TX, LcdBaud, ["Cal. Flag = ", DEC  Pos_Vars]
SEROUT TX, LcdBaud, [LcdCR]
 ' lcd & bp loop
DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))


FOR Word_Var = BaseR TO FinishR STEP 8
 GET  Word_Var,      Word RedLow
 GET (Word_Var + 2), Word RedHigh
 GET (Word_Var + 4), GreenLow
 GET (Word_Var + 5), GreenHigh
```

```
 GET (Word_Var + 6), BlueLow
 GET (Word_Var + 7), BlueHigh
 DEBUG CR, "RedLow   = ", DEC4 RedLow,  "     RedHigh = ", DEC4 RedHigh, "
Counter = ",DEC3 Word_Var, CR
 DEBUG "GreenLow = ", DEC4 GreenLow, "   GreenHigh = ", DEC4 GreenHigh, CR
 DEBUG "BlueLow  = ", DEC4 BlueLow, "    BlueHigh = ", DEC4 BlueHigh, CR
 ' lcd & bp loop
 SEROUT TX, LcdBaud, [LcdCls]
 PAUSE 250
 SEROUT TX, LcdBaud, ["Cnt = ", DEC3 Word_Var, " High  Low"]
 SEROUT TX, LcdBaud, [LcdCR]
 SEROUT TX, LcdBaud, ["Red       ", DEC4 RedHigh, "  ", DEC4 RedLow]
 SEROUT TX, LcdBaud, ["Blue      ", DEC4 BlueHigh, "  ", DEC4 BlueLow]
 SEROUT TX, LcdBaud, ["Green     ", DEC4 GreenHigh, "  ", DEC4 GreenLow]
 DO
    PAUSE 1
 LOOP WHILE ((BUTTON_1 = 0))
NEXT
'   lcd
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Done with Read"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Main Menu hit 1"]
' debug window
DEBUG CR, "     Done!", CR
DEBUG "Main Menu press Enter", CR
DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))


Main_Menu:                          'Loading last slot info for use in main
menu
PUT SPRAM_Lastslot, Read_pgm
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 5

RUN Menu_pgm
END
```

Init EEPROM Program listing

```
'{$STAMP BS2e}
'{$PBASIC 2.5}
'
'************************************************************
'  Program: Init_EEPROM  (utility program)
'  Purpose: Utility program used to quickly calibrate the servo arm
'           with "hard" values as needed
'  Authors: Rick Johnson & Chris Hileman
'  Date:    January 2004
'  Modified by: Rick Johnson & Luke Jamison
```

```
'               April 2007
'*****************************************************************
TX              PIN     9               ' serial output to LCD

    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32
LcdBaud         CON     T19K2

LcdBkSpc        CON     $08             ' move cursor left
LcdRt           CON     $09             ' move cursor right
LcdLF           CON     $0A             ' move cursor down 1 line
LcdCls          CON     $0C             ' clear LCD (use PAUSE 5 after)
LcdCR           CON     $0D             ' move pos 0 of next line
LcdBLon         CON     $11             ' backlight on
LcdBLoff        CON     $12             ' backlight off
LcdOff          CON     $15             ' LCD off

Menu_pgm        CON       0             ' Program slot numbers
Sort_pgm        CON       1
Cal_pgm         CON       2
Read_pgm        CON       3
Init_pgm        CON       4

Button_1        PIN      10             'pushbutton pins
Button_2        PIN      13
Button_3        PIN      14
Button_4        PIN      15

Home_Pos        VAR   Word            'Home_pos = Home position of Servo
Sensor_Pos      VAR   Word            'Sensor_Pos = Position of the RGB
Sensor
Hole_1          VAR   Word            'Pos_1 = Position of the #1 Tube
Pos_Delta       VAR   Word            'Delta of the Position of the tube
answer          VAR   Byte
temp            VAR   Byte
Mem_Loc         VAR   Byte

SPRAM_Lastslot  CON   62
Cal_flag        CON   8

Home_Pos   = 130
Sensor_Pos = 280
Hole_1     = 490
Pos_Delta  = 80
menu:
DEBUG "What would you like to do? ", CR
DEBUG "[1] Run Init prog", CR
DEBUG "[2] Main Menu", CR
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Menu Options: "]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Run Init prog hit 1"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Main Menu hit 2"]
```

```
DO
  PAUSE 1
LOOP WHILE ((BUTTON_1 = 0) AND (Button_2 = 0))

IF (Button_1 = 1) THEN
    GOTO Start
ELSEIF (Button_2 = 1) THEN
    GOTO Main_Menu
ENDIF


Start:
DEBUG "Stores Variables into EEPROM", CR

WRITE 0, Word Pos_Delta
WRITE 2, Word Hole_1
WRITE 4, Word Sensor_Pos
WRITE 6, Word Home_Pos

DEBUG "Done Writing to EEPROM!", CR
Pos_Delta  = 0
Hole_1     = 0
Sensor_Pos = 0
HOME_Pos   = 0

READ 0, Word Pos_Delta
READ 2, Word Hole_1
READ 4, Word Sensor_Pos
READ 6, Word Home_Pos

DEBUG "Pos_Delta  = ", DEC3 Pos_Delta, CR
DEBUG "Hole_1     = ", DEC3 Hole_1, CR
DEBUG "Sensor_Pos = ", DEC3 Sensor_Pos, CR
DEBUG "Home_Pos   = ", DEC3 Home_Pos, CR
DEBUG "Do these numbers make sense?  If not fix program!", CR
DEBUG "Main Menu press Enter", CR

SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
SEROUT TX, LcdBaud, ["Pos_Delta  = ", DEC3 Pos_Delta]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Hole_1     = ", DEC3 Hole_1]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Sensor_Pos = ", DEC3 Sensor_Pos]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Hit 1 to continue"]
DO
   PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))

DEBUG " ", CR
DEBUG "Home ", DEC3 Home_Pos, CR
DEBUG "If ", CR
DEBUG " ", CR

SEROUT TX, LcdBaud, [LcdCls]
PAUSE 250
```

```
SEROUT TX, LcdBaud, ["Home_Pos = ", DEC3 Home_Pos]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["If #'s are wrong"]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["fix program."]
SEROUT TX, LcdBaud, [LcdCR]
SEROUT TX, LcdBaud, ["Main Menu hit 1"]

DO
   PAUSE 1
LOOP WHILE ((BUTTON_1 = 0))

FOR Mem_Loc = 0 TO 7                ' read EEPROM, writing Scratchpad before
Main Menu
    READ Mem_Loc, TEMP
    PUT  Mem_Loc, TEMP
NEXT

GET Cal_flag, temp
IF temp <> 199 THEN                 'only update calibration flag if not
already calibrated
   PUT Cal_flag, 99
   WRITE Cal_flag, 99
ENDIF


Main_Menu:                          'Loading last slot info for use in main
menu
PUT SPRAM_Lastslot, Init_pgm
SEROUT TX, LcdBaud, [LcdCls]
PAUSE 5

RUN Menu_pgm
END
```